Generating Lists

MARCH 10TH, 2014
Generating lists

- Python has a built-in function called `range` that allows us to generate lists using *arithmetic progressions*.

- It can have one, two, or three arguments, all of which must be integers.

```python
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(3, 11)
[3, 4, 5, 6, 7, 8, 9, 10]
>>> range(0, 30, 5)
[0, 5, 10, 15, 20, 25]
>>> range(0, 10, 3)
[0, 3, 6, 9]
>>> range(0, -10, -1)
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
>>> range(0)
[]
>>> range(1, 0)
[]
```
The range function is useful in for-loops

```
for i in range(1, 10, 2):
    print i*i
```

- Repeats the execution of the body of the for-loop for each value of $i = 1, 3, 5, 7, \text{and} \ 9$.
- Equivalent to
  ```
i = 1
while i < 10:
    print i*i
    i = i + 2
  ```
- But more convenient for simple loops because no need to initialize before loop and no need to update within loop.
More examples of for-loops

```python
L = ["hello", "hi", "bye"]
for e in L:
    print e + e

s = "What is this sentence?"
for ch in s:
    print ch
```
Here is another useful way of generating lists, particularly for initializing them, i.e., assign them “initial” values at the start of a program.

Example:

\[ n = 25 \]
\[ L = [8] \ast n \]

This assigns to L a list of length 25 consisting of the integer 8.
Accessing slices of lists and strings

L = ["hi", 10, "bye", 100, -20, 123, 176, 3.45, 1, "it"]

Examples:

• L[2:5] is ["bye", 100, -20]
• L[:2] is ["hi", 10]
• L[4:4] is []
• L[4] = -20
• L[:len(L):2] = ["hi", "bye", -20, 176, 1]
• L[2:5][1] = 100
• L[1:5][:2] = [10, "bye"]
Slice Notation

- The basic notation
  \[L[start:end]\] # sublist with items indexed start through end - 1
  \[L[start:]\] # sublist with items indexed start through end of list
  \[L[:end]\] # sublist with items from the start of the list through index end-1
  \[L[:]\] # a copy of the original list

- The notation can also be used with a third parameter, step.
  \[L[start:end:step]\] # sublist with items indexed start, not past end, in increments of step

- Step can also be negative, in which case the elements are listed in reverse order
Problem

- Read a positive integer $n$ and roll two $n$-sided dice a million times and output the distribution of the sums.

- In other words,
  - the number of times 2 appears as the sum,
  - the number of times 3 appears as the sum,
  - the number of times 4 appears as the sum,
  - ...
  - the number of times $2n$ appears as the sum.
import random

n = int(input("Please type the number of sides in your dice."))

L = [0]*(2*n+1)  # Creates a list of length 2*n+1 with all elements of the
               # list initialized to 0

for i in range(1000000):
    # Roll the two n-sided dice and record the outcome
    outcome = random.randint(1, n) + random.randint(1, n)

    # L[outcome] stores the number of times outcome has appeared
    # So this element in the list needs to be incremented
    L[outcome] = L[outcome] + 1

#Report the contents of slots 2, 3, ...
print L[2:]